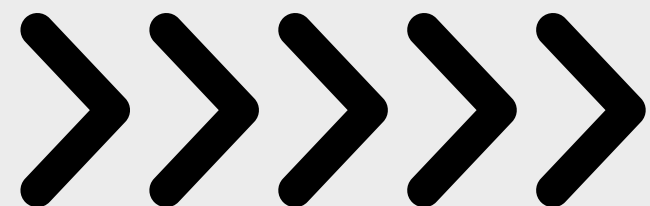


TESTES

UNITÁRIOS E INTEGRAÇÃO



Prof. Thiago Suzuqui Lodi





TESTES UNITÁRIOS

- **O que são**

- Testes que verificam o funcionamento de unidades isoladas do código:
 - funções, métodos ou classes;
- Para garantir que cada peça faz exatamente o que deveria, independente do resto do sistema.

- **Vantagens**

- Detecção rápida de erros em partes isoladas do sistema;
- Servem como documentação viva do comportamento esperado;
- Facilitam refatoração se os testes passam, você tem confiança na mudança.



- **Limitações**

- Não garantem que os módulos estão funcionando em conjunto;
- Podem falhar em detectar problemas de uso real:
 - dependências externas, performance, tempo de resposta.





TESTES UNITÁRIOS

- **Quem deve fazer?**
 - Espera-se que cada Dev escreva os testes unitários do que desenvolveu;
 - Ele é quem conhece a lógica e os possíveis pontos frágeis;
 - Ajuda a garantir que futuras alterações não quebrem funcionalidades;
 - Evita sobrecarregar o QA/Tester.
- 
- 



PADRÃO AAA

- **Arrange**
 - preparar o cenário de teste
 - variáveis, instâncias, mocks
- **Act**
 - executar o teste
 - função ou método
- **Assert**
 - validar o resultado obtido
 - retorno, variáveis alteradas, mocks executados



EXEMPLO JAVA



```
1  public class Calculadora {  
2      public int soma(int a, int b) {  
3          return a + b;  
4      }  
5  
6      public int subtrai(int a, int b) {  
7          return a - b;  
8      }  
9  }
```



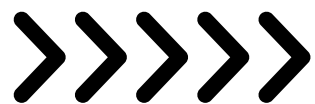
x x x x



EXEMPLO JAVA



```
1  import org.junit.jupiter.api.Test;
2  import static org.junit.jupiter.api.Assertions.*;
3
4  public class CalculadoraTest {
5      Calculadora calc = new Calculadora();
6
7      @Test
8      void somaTest() {
9          int a = 2;
10         int b = 3;
11
12         int resultado = calc.soma(a, b);
13
14         assertEquals(5, resultado);
15     }
16
17     @Test
18     void subtraiTest() {
19         int a = 5;
20         int b = 3;
21
22         int resultado = calc.subtrai(a, b);
23
24         assertEquals(2, resultado);
25     }
26 }
```



x x x x



EXEMPLO PHP



```
1  <?php
   0 references | 0 implementations
2  class Calculadora {
   0 references | 0 overrides
3      public function soma($a, $b): mixed {
4          return $a + $b;
5      }
6
   0 references | 0 overrides
7      public function subtrai($a, $b): float|int {
8          return $a - $b;
9      }
10 }
```



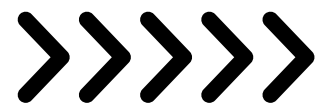
x x x x



EXEMPLO PHP



```
2 use PHPUnit\Framework\TestCase;
3
4 require 'Calculadora.php';
5
6 0 references | 0 implementations
class CalculadoraTest extends TestCase {
7     0 references
    private $calc;
8
9     0 references | 0 overrides
    protected function setUp(): void {
10         $this->calc = new Calculadora();
11     }
12
13     0 references | 0 overrides
    public function somaTest(): void {
14         $a = 2;
15         $b = 3;
16
17         $resultado = $this->calc->soma($a, $b);
18
19         $this->assertEquals(5, $resultado);
20     }
21
22     0 references | 0 overrides
    public function subtraiTest(): void {
23         $a = 5;
24         $b = 3;
25
26         $resultado = $this->calc->subtrai($a, $b);
27
28         $this->assertEquals(2, $resultado);
29     }
30 }
```



x x x x



EXEMPLO JS



```
1  function soma(a, b) {  
2    |   return a + b;  
3  }  
4  
5  function subtrai(a, b) {  
6    |   return a - b;  
7  }  
8  
9  module.exports = { soma, subtrai };  
  |
```



x x x x



EXEMPLO JS



```
2  test('soma números corretamente', () => {
3      const a = 2;
4      const b = 3;
5
6      const resultado = soma(a, b);
7
8      expect(resultado).toBe(5);
9  });
10
11 test('subtrai números corretamente', () => {
12     const a = 5;
13     const b = 3;
14
15     const resultado = subtrai(a, b);
16
17     expect(resultado).toBe(2);
18 });
```



x x x x





FRAMEWORKS

- Java – JUnit
- PHP – PHPUnit
- JavaScript – Jest
- Todos seguem padrões similares funções, mocks, asserts.

The JUnit logo, featuring the word "JUnit" in a serif font. The "J" is green and the "Unit" is red.The PHPUnit logo, featuring the word "PHPUnit" in a sans-serif font. The "P" and "H" are blue, the "P" is green, and the "Unit" is blue.



PRÁTICA

Certifique-se que tenha o Node.js instalado <https://nodejs.org/pt>

Clone o repositório <https://gitlab.com/senachubacademy/146/testes-js>

Na raiz do projeto execute o comando **npm install**

Para executar os testes utilize o comando **npm run test**

Para criar os testes vamos utilizar a lib Jest <https://jestjs.io/>

A partir das classes Calculadora.js e Boletim.js defina os casos de teste e implemente os testes unitários, de acordo com o que vimos nas aulas anteriores.

